Last time…

We discussed

- Recursion
- Tree data structures

A number of people falling behind on the Labs

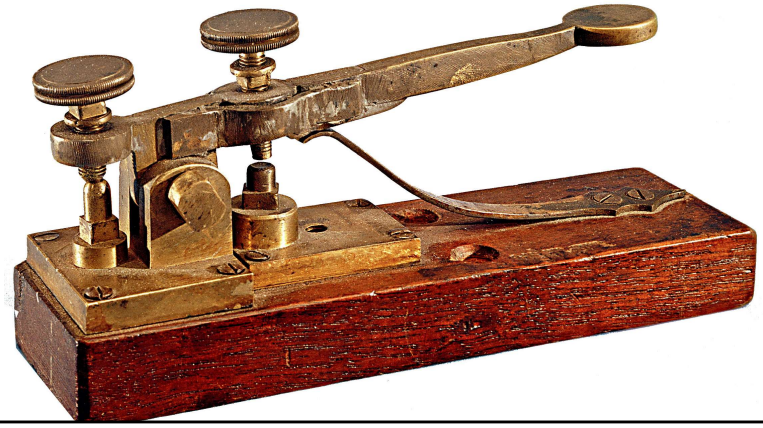
- Labs are crucial building blocks for Project02, for learning course material
- Contact me if you want to catch up so we can make a plan

QUIZ#5 problem review in Eclipse

CS112 –
Java
Programming

Fall 2022

Project 02 – Huffman Coding



A TELEGRAPH KEY

Data Compression

What?

Why?

How?

REPRESENT DATA USING A REPRESENTATION THAT TAKES FEWER BITS THAN THE "natural" REPRESENTATION
SAVES STORAGE SPACE, SAVES TRANSMISSION TIME OVER A DATA NETWORK
Two main "flavors": lossless, lossy
Why lossy? MUCH more compression. Usually for audio and video, where human eyes/ears/brain are happy with "near-perfect" representation
EXAMPLES of Lossless: WinZip, WinRAR, 7-zip. Take advantage of statistics of input: achieve 2x-3x compression
Lossy: MPEG, used in DVDs, AVC, used in Blu-ray, other newer formats used for internet video streaming. Achieve 500-1000x compression for HDTV.
My whole pre-teaching career in media compression, a multi-billion dollar industry.
DVD stores 25 seconds of uncompressed HDTV. 3 hours of compressed.

Compression Origins

Morse Code

A	.–	M	––	Y	–.––
B	–...	N	–.	Z	––..
C	–.–.	O	–––	1	.––––
D	–..	P	.––.	2	..–––
E	.	Q	––.–	3	...––
F	..–.	R	.–.	4	....–
G	––.	S	...	5	.....
H	....	T	–	6	–....
I	..	U	..–	7	––...
J	.–––	V	...–	8	–––..
K	–.–	W	.––	9	––––.
L	.–..	X	–..–	0	–––––

What's the problem with Morse Code?

You know the distress signal "SOS"? Know why it is the distress signal? B/c Morse Code rep'n is easy pattern: …---…
Morse code used in telegraph. Telegraph probably biggest communication revolution in history, except maybe for printing press
What is Morse code? Patterns of dots and dashes, to represent English letters and digits 0-9
What's the magic? Common letters have short codes, uncommon letters have long codes
Avg length = sum (prob(each char) * len(each char))

Huffman Codes – Morse for Computers

Used in MPEG-1 and MPEG-2 video compression standards

Newer version used in newer audio and video compression standards

For Project 02, you will implement a classic Huffman Code

Only two symbols in the “coded alphabet”:

- Usually called “0” and “1”

What is a standard? A detailed specification that lets one system perform compression and a separate system perform decompression, with the "correct" results.

Symbols and Alphabets

	Alphabet	**Symbols**
English text	abcdefghijklmnopqrstuvwxyz	Could be LETTERS (i.e. same as alphabet) Could be WORDS (cat, run, chair, keyboard, …)
Morse Code	"dot", "dash", "space"	.- ('a') -… ('b') -.-. ('c'), etc
Huffman Code	0, 1	Discuss next!

"alphabet" is set of all characters used to construct symbols
"symbols" is set of all legal patterns of characters, and associated meanings

Huffman Codes

How can Huffman Codes use only two symbols while Morse Codes need three?

Prefix-free set of symbols

Input	Compressed Symbol
α	00
β	01
γ	100
δ	101
ε	11

No codeword is a PREFIX of any other code. Not true for Morse.
Walk thru how to decode.
Construction of Huffman codes ensures prefix-free symbol set

Huffman Codes

Construction of Huffman Codes

Need to define our input symbol set

Need the probability of occurrence of each symbol in the set

Construction recipe…

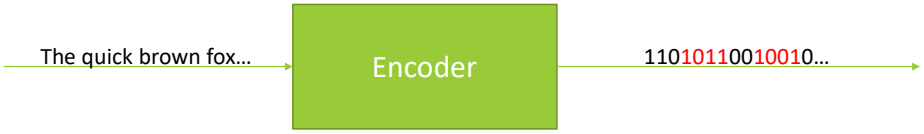
Well, given our past few weeks of lectures, Huffman Code construction and use will require advanced data structures such as lists and trees, and also recursion!

CONSTRUCTION:
- DEFINE A CLASS THAT STORES 'meaning' AND 'symbol' and 'probability' FOR A PARTICULAR SYMBOL
- MAKE A LIST OF ALL OBJECTS, ONE PER SYMBOL. Sort by increasing probability (count)
- Start combining into tree elements: need to extend the class
- Keep sorting and combining, until only one element in the list, which is root of the tree

Now we assign symbols: '0' for left branch, '1' for right branch. Recursively!
VERIFY NO-PREFIX PROPERTY!

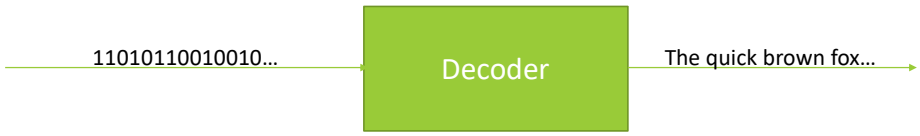
Encoding with Huffman Codes

Coded symbol (pattern of 0's and 1's) for each *input symbol* (text character, in our case)



Decoding with Huffman Codes

Want the Huffman tree…



The Project!

Huffman Project

Three programs:

- **Generate.java** – generate a Huffman table
- **Encode.java** – use Huffman table to compress a text inputfile
- **Decode.java** – use Huffman table to decompress output from **Encode.java**

Details:

- Set of *input symbols* is a small subset of UTF-16, a little bigger than ASCII
  - '\u0004' = End of Transmission
  - '\u0007' to '\u00fe' inclusive. Standard ASCII plus many other useful characters
  - **Generate** and **Encode** should ignore any input characters outside of this set

Generate:
Encode: output should be text file of '0' and '1' characters, not actually binary bits. NO SPACES, NO NEWLINES.
Decode: input is file of '0's and '1's, output is text

WRITE SHORT PROGRAM TO PRINT ASCII VALUES AS CHARACTERS!!

Details

Think about how you will test. Must write and turn in a test doc!

Think about design (data structures, sequence of operations) for each program

Think about code reuse. You have several pieces already from the Labs

Grading:

- Automated test cases, as usual
- 1 page test document, like last time
- Instructor/TA evaluation of design and SW quality, as usual

NOW, ON TO THE LAB!